Optimization of Production Line Scheduling: An OGA case study

Carmen Linares Vázquez[1][0009−0006−8266−3458], Julia de Pineda Gutiérrez[1][0000−0003−3416−4209], Nuria Gómez-Vargas[1][0000−0002−5051−9367], and Claudia Rodríguez-Vegas*[1][0009−0008−8968−4555]

Department of Artificial Intelligence and Operational Research, OGA, Seville, Spain
{nuria.gomez,carmen.linares,julia.pineda,claudia.rodriguez}@oga.ai
Corresponding author: *claudia.rodriguez@oga.ai

Abstract. This paper addresses a real industrial advanced planning and scheduling problem modeled as an Unrelated Parallel Machine Scheduling Problem (UPMSP) with sequence-dependent setup time. The objective of the tackled UPMSP is to generate feasible production plans that balance service level requirements, mainly through lateness minimization, with operational efficiency in terms of changeover costs, production volume, and line utilization. The case study involves multiple production lines, detailed working calendars, resource unavailability, material restrictions, shared resources, and alternative production versions, making feasibility a central challenge. To solve the problem, we propose a heuristic framework that preserves feasibility and combines a constructive phase, which builds schedules satisfying the operational constraints, with an improvement phase based on Large Neighborhood Search (LNS) and Variable Neighborhood Descent (VND) procedures. The results on real manufacturing instances show that the proposed method produces high quality, operationally meaningful schedules over long planning horizons within competitive computational times.

Keywords: Unrelated parallel machines · Scheduling · Metaheuristics

1 Introduction and Motivation

Production planning is a highly relevant and widespread problem in the manufacturing industry. In its simplest formulation, it can be addressed as a scheduling problem in which production tasks must be sequenced over time across different resources while optimizing criteria such as makespan, resource utilization, or tardiness minimization [6]. However, real industrial environments introduce additional complexity through operational constraints, resource availability, and changing business rules. In addition, production plans often require continuous re-planning to respond to disruptions such as machine breakdowns, supply delays, or demand fluctuations [7]. As a result, the problem becomes highly combinatorial, dynamic and context-dependent, making it difficult to solve with purely exact approaches in large-scale instances [5].

From an academic perspective, production planning and scheduling problems have been addressed using various optimization techniques [1]. Exact methods, such as mixed-integer linear programming (MIP), provide precise mathematical formulations but face scalability limitations. In contrast, heuristic and metaheuristic approaches are often more suitable for large scale industrial applications, as they can provide high quality solutions within short runtimes.

This work is motivated by the needs of a large Spanish agri-food manufacturer producing, packaging and distributing oil products across multiple plants. The planning system had to be deployable across facilities with different line capabilities, calendars, resource availability and material restrictions. The problem is an UPMSP variant with sequence-dependent setups, shared resources, routing flexibility and material constraints, where line assignment, sequencing, and resource usage are strongly interdependent [2].

This research is conducted within the framework of the Prodplanning R&D project, entitled "Production Planning and Sequencing in the Manufacturing Industry Based on Metaheuristic and Matheuristic Techniques". The project, led by Qosit Consulting S.L. (OGA), aims to develop a technological solution to improve efficiency in industrial production planning. In this context, it seeks to bridge the gap between theoretical scheduling models and the operational requirements encountered in real manufacturing environments. The main contribution is a general hybrid framework that combines constructive procedures, improvement heuristics, and advanced search techniques to balance solution quality and computational time. Its design supports deployment across different plants without algorithm redesign and enables rapid re-planning scenarios.

The remainder of this paper is organized as follows. Section 2 describes the industrial case study, Section 3 presents the proposed heuristic framework, Section 4 discusses the computational results, and Section 5 concludes the paper and outlines future research.

2 Case Study

This section presents the industrial case study, its main constraints, and the optimization criteria.

2.1 Industrial Context and Problem Definition

From a modeling perspective, the planning problem is defined over a finite horizon $\mathcal{T}$, a set of plants $\mathcal{P}$, and production lines $\mathcal{L} = \bigcup_{p \in \mathcal{P}} \mathcal{L}_p$. A set of demands $\mathcal{D}$ must be scheduled, where each demand $d \in \mathcal{D}$ is characterized by a material $m(d)$, a quantity q_d, an earliest production date e_d, a delivery date u_d, and a priority level π_d. Let $\mathcal{M}$ denote the set of materials. Each material $m \in \mathcal{M}$ is described by an attribute vector a_m, including all characteristics (product type, format...) that affect compatibility constraints and changeover requirements. In addition, each material has an associated bill of materials $\mathcal{B}_m$, including the raw

and auxiliary materials required for its production, and a set of lines $\mathcal{L}(m) \subseteq \mathcal{L}$ where it can be produced.

Each production line $\ell \in \mathcal{L}$ is characterized by speeds $v_{\ell m}$, which depend on the attribute vector a_m of each material, and sequence dependent changeovers. Let $\tau_{\ell mm'}$ and $c_{\ell mm'}$ denote, respectively, the duration and cost of switching line ℓ from material m to material m'. Lines may also require secondary resources, such as shared equipment. Let $\mathcal{R}_{\ell m}$ denote the set of secondary resources required to produce material m on line ℓ. Finally, let $\nu_{\ell m} = (\mathcal{B}_m, \mathcal{R}_{\ell m})$ denote the production version associated with bill of materials B_m and routing $\mathcal{R}_{\ell m}$.

Line availability is represented through feasible working intervals $\mathcal{S}_\ell$, which incorporate shifts, breaks and plant calendars. A feasible solution assigns a subset of demands to compatible lines and schedules them over time, determining for each scheduled demand $d(m)$ a line $\ell(d(m)) \in \mathcal{L}(m)$, a start time s_d and a completion time C_d, while satisfying line calendars $\mathcal{S}_\ell$, material availability, shared resource restrictions and the auxiliary operations induced by the sequence.

The central trade-off is therefore between timely deliveries and operational efficiency. Grouping similar products reduces changeovers and extends equipment lifespan, but may conflict with delivery priorities.

2.2 Operational Constraints

The operational constraints derived from this industrial environment significantly extend the classical parallel machine scheduling formulation. These constraints, arising from plant limitations and business rules, make feasibility depend not only on local sequencing decisions within each line, but also on global interactions across lines.

Temporal constraints. Each demand $d \in \mathcal{D}$ has a hard earliest production date e_d, and a soft delivery date u_d:

$$s_d \geq e_d. \tag{1}$$

Late production with respect to u_d is allowed but penalized according to the priority level π_d.

Working calendars and shifts. Production is restricted to feasible working intervals. For each line $\ell \in \mathcal{L}$, let $\mathcal{S}_\ell$ denote the set of feasible time intervals determined by shifts, working hours, and planned stoppages. Then, for any scheduled demand d, its production interval must satisfy:

$$[s_d, C_d] \subseteq \bigcup_{I \in \mathcal{S}_{\ell(d)}} I. \tag{2}$$

Material-line compatibility. Each demand d can only be assigned to lines compatible with its material:

$$\ell(d) \in \mathcal{L}(m(d)). \tag{3}$$

Shared resources and line coupling. Let $\mathcal{R}$ denote the set of secondary resources, shared across multiple lines, and κ_r denote the maximum number of simultaneous uses of resource r. Resource usage must satisfy capacity constraints at any time $t \in \mathcal{T}$:

$$\sum_{\substack{d \in \mathcal{D} \\ t \in [s_d, C_d]}} \mathbf{1}\{r \in \mathcal{R}_{\ell(d),m(d)}\} \leq \kappa_r, \quad \forall r \in \mathcal{R}. \tag{4}$$

Additionally, groups of lines may be subject to mutual exclusion constraints, preventing the simultaneous production of materials with specific characteristics.

Sequence-dependent changeovers and auxiliary operations. The production sequence must include auxiliary operations such as pre-start, cleaning, and closing events. These operations take time and must be scheduled consistently with the production sequence and the working calendar. Moreover, changeovers introduce sequence-dependent time and cost penalties, which depend on the ordered material pair (m, m').

Resource and material unavailability. Lines, resources, and materials may be unavailable during certain time intervals. Let $\mathcal{U}_x \subseteq \mathcal{T}$ denote the set of unavailability intervals for element x (line, resource, or material). Then, any production requiring x must satisfy:

$$[s_d, C_d] \cap \mathcal{U}_x = \emptyset.$$

These constraints capture maintenance periods, breakdowns, or lack of materials.

2.3 Optimization Criteria

The industrial planning problem is inherently multiobjective. In classical production scheduling [4], criteria include minimizing makespan, tardiness, machine idle time, or setup costs. However, we consider a criterion that combines service-level requirements, production volume, schedule compactness and operational efficiency, as summarized in Figure 1. These objectives are sequenced lexicographically according to business rules, while demand priorities add flexibility within this hierarchy by allowing planners to express different urgency levels without using weighted objective functions, whose impact is difficult to anticipate across heterogeneous instances.

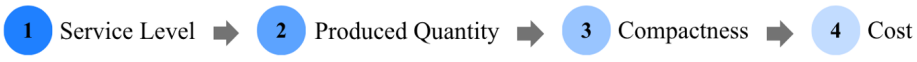


Fig. 1. Lexicographic ordering of the optimization criteria.

Let $\mathcal{D}^{\text{sp}} \subseteq \mathcal{D}$ be the special-priority demands and $\mathcal{D}^{\text{sch}} \subseteq \mathcal{D}$ the scheduled demands in a given solution. For each demand $d \in \mathcal{D}^{\text{sch}}$, let C_d be its completion time and u_d its delivery date. The first objective maximizes the number of

scheduled special-priority demands, or equivalently

$$\min \quad -\left|\mathcal{D}^{\text{sch}} \cap \mathcal{D}^{\text{sp}}\right|. \tag{5}$$

Then, delay is minimized following the priority hierarchy of the orders. The delay of demand d is defined in days as $T_d = \max\{0, C_d - u_d\}$.

For any priority class $\mathcal{D}^k \subseteq \mathcal{D}$, the corresponding delay component is written as

$$\min \quad \sum_{d \in \mathcal{D}^{\text{sch}} \cap \mathcal{D}^k} T_d. \tag{6}$$

They are evaluated sequentially, from special-priority demands to the remaining ones with relevant priorities. This encourages the most critical orders to be scheduled as early as possible. In contrast, the lowest priority class of demands is treated with lower urgency. Let π^{low} denote the lowest priority level, and define $\mathcal{D}^{\text{low}} = \{d \in \mathcal{D} : \pi_d = \pi^{\text{low}}\}$. Its objective is to reduce delayed orders

$$\min \quad \sum_{d \in \mathcal{D}^{\text{sch}} \cap \mathcal{D}^{\text{low}}} \mathbf{1}\{C_d > u_d\}. \tag{7}$$

This design prevents an already delayed low-priority order from driving the overall schedule. Such demands are sequenced only when the operational impact is minimal, rather than forcing additional costly transitions. After delay-related criteria, the algorithm seeks to maximize total quantity produced.

$$\min \quad -\sum_{d \in \mathcal{D}^{\text{sch}}} q_d. \tag{8}$$

A compactness criterion is also introduced to avoid artificially sparse plans. Without this term, low priority demands may be postponed to future compatible blocks only to reduce changeover cost. We consider an artificial gap to be an idle interval caused by a demand with an earliest production date or a material unavailability, rather than by a specific line stoppage, so such gaps are potentially usable by other compatible demands. This distinction is relevant because a classical idle-time objective would only reduce the total amount of idle time, regardless of whether the gap is filled before or after the blocking demand, whereas the proposed criterion explicitly favors earlier use of available capacity.

$$\min \quad N^{\text{post}}, \tag{9}$$

where N^{post} counts delayed demands postponed by these artificial-gap events.

The next objective is the total changeover cost. For each line $\ell \in \mathcal{L}$, let $\sigma_\ell = (d_1, \ldots, d_{n_\ell})$ denote the sequence of scheduled demands assigned to that line. The cost of changing from material m to material m' on line ℓ combines cost associated to the duration of the changeover, the cost of wasted raw material, and any cleaning cost:

$$\text{ChCost}_{\ell m m'} = \tau_\ell \cdot \mathit{IdleTime} + \mathit{WastMatl}_{mm'} \cdot \mathit{Mat} + \mathit{Clean}_{mm'}. \tag{10}$$

All terms are expressed in euros: $\tau_\ell IdleTime$ accounts for line idle cost, $WastMatlmm'Mat$ for wasted material, and $Cleanmm'$ for cleaning cost when required. Therefore, the objective is formulated as

$$\min \quad \sum_{\ell \in \mathcal{L}} \sum_{i=1}^{n_\ell - 1} \mathrm{ChCost}_{\ell, m(d_i), m(d_{i+1})}. \tag{11}$$

Placing this criterion later in the hierarchy avoids schedules that reduce transitions by excessively delaying or discarding demands.

3 Solution: A Multi-Start Constructive and Improvement Heuristic

The proposed approach is a multistage heuristic. A multi-start constructive phase first builds a feasible solution mainly driven by tardiness minimization (6). A Large Neighborhood Search (LNS), a metaheuristic based on the iterative destruction and repair of the solution, then improves allocation and sequencing decisions, followed by a Variable Neighborhood Descent (VND) procedure using different neighborhood moves to reduce operational inefficiencies, especially changeover cost (11), while preserving service levels.

Let $x \in \mathcal{X}$ be a feasible solution and let $\mathcal{O}(x)$ be the set of production orders scheduled in x. For each line $\ell \in \mathcal{L}$, the solution is represented as a time ordered sequence

$$\mathcal{O}_\ell(x) = (o_{\ell,1}, \ldots, o_{\ell,n_\ell}),$$

where each production order $o_{\ell,i} \in \mathcal{O}(x)$ is associated with a demand $d(o)$ and a production version $\nu_{\ell m}(o)$. Each order o is described by a set of time intervals

$$\mathcal{I}_o = \mathcal{I}_o^{\mathrm{pre}} \cup \mathcal{I}_o^{\mathrm{chg}} \cup \mathcal{I}_o^{\mathrm{prod}} \cup \mathcal{I}_o^{\mathrm{cln}} \cup \mathcal{I}_o^{\mathrm{cls}},$$

corresponding respectively to pre-start, changeover, productive, cleaning, and closing operations, as illustrated in Figure 2. The auxiliary intervals may be empty when the corresponding operation is not required, whereas $\mathcal{I}_o^{\mathrm{prod}}$ must be non-empty for every scheduled order. Then the demand start and completion times are induced by the intervals of the order:

$$s_{d(o)} = \min\{a : [a,b) \in \mathcal{I}_o\}, \qquad C_{d(o)} = \max\{b : [a,b) \in \mathcal{I}_o\}. \tag{12}$$

All modifications are performed through insertion and removal operators that preserve feasibility. Let $\mathrm{Ins}(x, d, \ell, \nu)$ denote the operator that attempts to insert demand d on line ℓ using production version ν, and let $\mathrm{Rem}(x, o)$ denote the removal of an already scheduled order o. The operator $\mathrm{Ins}(x, d, \ell, \nu)$ computes feasible time windows $\Omega_\nu(x)$ and attempts to build a complete order within them, first consecutively after the last order of the line and, otherwise, at the earliest feasible position. The insertion fails only if no such position can accommodate the complete order.

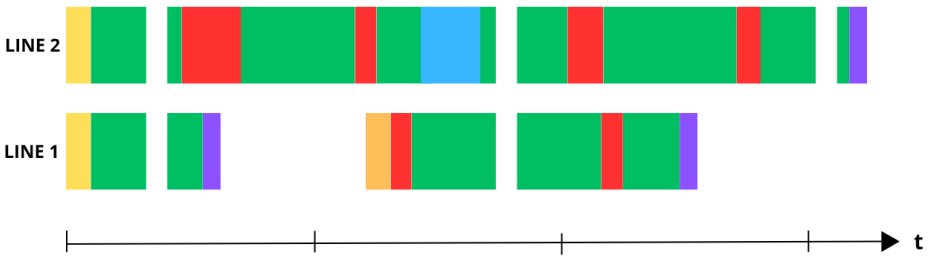


Fig. 2. Gantt representation of the solution. Colors indicate interval types: productive (green), changeover (red), pre-start (yellow), cleaning (blue), and closing (purple). Pre-start intervals may overlap with changeovers (orange), and idle periods (white) may occur due to operational constraints.

Shared resource feasibility is maintained through the global shared resource occupation denoted as $\Gamma(x)$. When an order is inserted or removed, the solution updates $\Gamma(x)$ by adding or releasing the occupation required by $\mathcal{I}_o$.

Production lines are modified according to a LIFO mechanism. Therefore, when a move affects an intermediate order $o_{\ell,i}$, the tail

$$T^{+}_{\ell,i}(x) = (o_{\ell,i+1}, \ldots, o_{\ell,n_\ell})$$

is removed and rebuilt using the same insertion operator. Removing orders from the end preserves the feasible prefix and avoids inconsistent intermediate schedules, since all orders after the affected position may have to be rescheduled, preserving feasibility and enabling compact rebuilding.

Line-closing operations are handled explicitly through $\mathcal{I}^{\text{cls}}_o$. Each newly inserted order is generated with a final closing interval. If a consecutive order is inserted, this closing may be removed; otherwise, it is restored and the corresponding resource occupation is recomputed.

Overall, this framework ensures that all constructive and improvement procedures operate over feasible schedules. This design is particularly important in highly constrained industrial environments, where feasibility depends on several operational competing requirements.

3.1 Constructive Phase

The constructive procedure starts by initializing the line sequences $\mathcal{O}_\ell(x)$ with the ongoing production orders from the previous execution and their shared resource occupation $\Gamma(x)$ and then builds an initial feasible solution $x^0 \in \mathcal{X}$. Let $\mathcal{D}^{\text{un}}(x) \subset \mathcal{D}$ be the set of unassigned demands in the solution x. At each iteration, the algorithm selects an active line ℓ^*. For each line ℓ, let

$$\mathcal{C}_\ell(x) = \{d \in \mathcal{D}^{\text{un}}(x) : \exists \nu_{\ell m}(d) \text{ and } \Omega_\nu(x) \neq \emptyset\} \tag{13}$$

be the set of unassigned demands that can be sequenced on line ℓ. Lines with feasible priority 7 demands are preferred, otherwise, the next line is selected

randomly among those with non-empty $\mathcal{C}_\ell(x)$. Once a line is activated, it stays open until the minimum operating time $\tau_{\min}^{\text{open}}$ is satisfied, unless no compatible demand remains. This prevents short and fragmented production periods.

For the active line ℓ^*, candidates in $\mathcal{C}_{\ell^*}(x)$ are sorted following a lexicographic criterion. The first key is the earliest feasible production date derived from $\Omega_{\nu(d)}(x)$. For delayed demands, the end of the material availability window is also considered to reduce future infeasibility. The rule then favors demands with a smaller changeover cost and penalizes formats scheduled on other lines. Delayed demands are considered firstly; among them, demands with $\pi_d = 7$ are sorted by delivery date u_d and then by changeover cost $c_{\ell^* m' m}$, with m' the last material produced in ℓ^*, whereas demands with $\pi_d = 10$ are sorted first by changeover cost and then by delivery date.

The first ranked demand d^* defines a batch $\mathcal{Q}$ of demands with the same material $m^* = m(d^*)$ and feasible production date. Delayed demands are inserted according to priority and delivery date. To avoid excessively long batches, non delayed demands are added only while their cumulative processing time remains below the threshold $\tau_{\max}^{\text{batch}}$:

$$\sum_{d \in \mathcal{Q} \cap \mathcal{D}^{\text{un}}} \frac{q_d}{v_{\ell^* m^*}} \leq \tau_{\max}^{\text{batch}}. \tag{14}$$

If a demand insertion turns out to be infeasible, the pair (d, ℓ^*) is temporarily marked as unavailable. Such demands can be reconsidered on the same active line only up to $N_{\max}^{\text{att}}$ attempts. Once the line is released, the mark is removed and the demand becomes eligible again for any compatible line.

The constructive phase is executed for N^{seed} random seeds. Then, the best resulting solution according to the lexicographic criteria, x^0, is passed to the improvement phase. x^0 contains the scheduled production orders $\mathcal{O}_\ell(x^0)$, the unassigned demands $\mathcal{D}^{\text{un}}(x^0)$, and the associated objective function metrics.

3.2 Improvement Phase

The improvement phase combines a LNS procedure and a VND with multiple local search moves, both implemented through the feasibility-preserving insertion and removal operators introduced above. The VND explores intra- and inter-line neighborhoods based on swap and reinsert moves, each of them also combined with 2-opt moves. The LNS phase iteratively applies a destroy and repair mechanism. At iteration k, a subset of scheduled orders

$$R^k \subseteq \mathcal{O}(x^k)$$

is removed from the current solution. Lines are then rebuilt using the constructive procedure, generating a new solution x^{k+1}. This method generates alternative feasible solutions, otherwise difficult to reach through local modifications.

Let x^{LNS} denote the best solution obtained after the LNS. At this stage, the solution typically achieves low tardiness but may exhibit high changeover

costs. To address this trade-off, an ε-constraint strategy is introduced. For each scheduled demand d with $\pi_d = 7$, a target delay is fixed as

$$\bar{T}_d = T_d\left(x^{\mathrm{LNS}}\right).$$

Subsequent local search moves are then evaluated under this ε-constraint strategy. In this stage, delay is not explicitly minimized in the objective function. Instead, any candidate solution x' is evaluated following a lexicographic criterion based on the remaining objectives, but they must always satisfy

$$T_d(x) \leq \bar{T}_d + \varepsilon_7, \qquad \forall d \in \mathcal{D} : \pi_d = 7, \tag{15}$$

where ε_7 is the allowed delay tolerance.

4 Results

The performance of the proposed approach is evaluated in a set of diverse real-world scenarios, where we assess the impact of key modeling features on the resulting schedules and optimization criteria defined in Section 2.3.

The main parameters were fixed for all experiments to ensure a consistent comparison across scenarios. The planning horizon was set to $\mathcal{T} = 28$ days, the constructive phase was run with $N^{\mathrm{seed}} = 24$ random seeds, and the minimum line opening time was fixed at $\tau^{\mathrm{open}}\mathrm{min} = 15$ h. The batch extension threshold was set to $\tau^{\mathrm{batch}}\mathrm{max} = 24$ h, while the lateness tolerance for priority 7 demands was defined as $\varepsilon_7 = 3$ days. All values correspond to business rules, except for N^{seed}, which was empirically calibrated.

The analysis focuses on how the solution framework can generate feasible and compact schedules. Gantt charts are used to demonstrate how the algorithm coordinates production activities competing for shared resources. Figure 3 shows a representative fragment of the schedule for the line group

$$G^{\mathrm{cap}} = \{\mathrm{L01}, \mathrm{L04}, \mathrm{L05}, \mathrm{L06}\},$$

which is subject to a maximum simultaneous capacity of $\kappa_{G^{\mathrm{cap}}} = 3$.

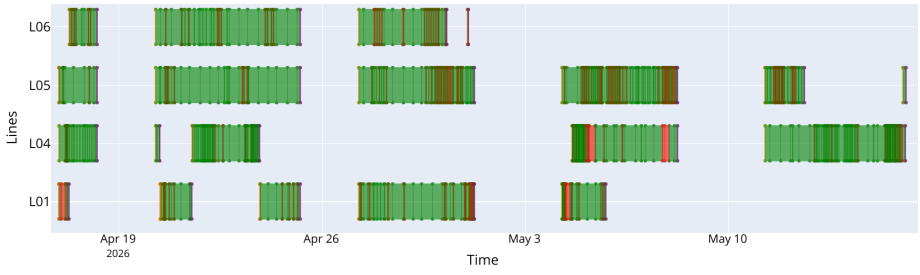


Fig. 3. Representative Gantt chart fragment for lines L01, L04, L05 and L06.

The planning horizon assumes continuous production, except during weekends and public holidays. In this case, May 1 and May 8 are non-working days as shown in Figure 3. It also illustrates feasibility in line with the shared capacity constraint. Although G^{cap} contains four lines, the generated schedule never activates more than three at a time, satisfying constraint (4) with $\kappa_{G^{\text{cap}}} = 3$.

Figure 4 highlights the effect of secondary resource coupling. Let $r^{\text{BOX}} \in \mathcal{R}$ denote a boxing machine, required by some production activities in lines L01 and L04, that is, $r^{\text{BOX}} \in \mathcal{R}_{\text{L01},m}$ or $r^{\text{BOX}} \in \mathcal{R}_{\text{L04},m}$. Since this resource has unit simultaneous capacity, $\kappa_{r^{\text{BOX}}} = 1$, orders on L01 and L04 overlap only when they do not simultaneously require r^{BOX}, showing that the secondary resource constraint (4) is correctly enforced.

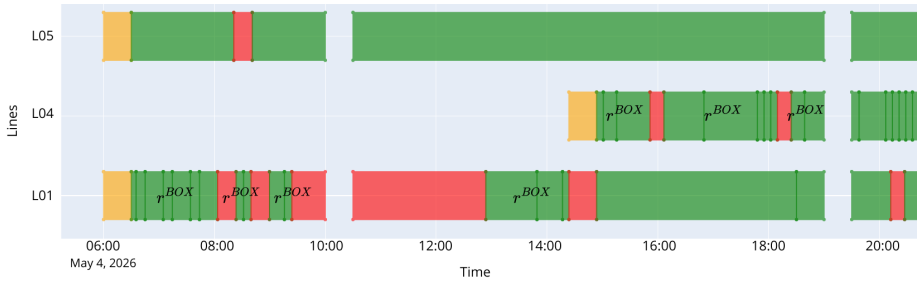


Fig. 4. Example of secondary resource feasibility in the schedule. Productive intervals labeled with r^{BOX} require the same boxing resource; therefore, line L04 only starts using this resource after line L01 has released it.

A second analysis evaluates the effect of demand priorities. In the baseline scenario, all demands have priority 10, so operational efficiency dominates lateness reduction. As shown in Figure 5, production on L04 is grouped into compact format-based batches, which is desirable because format changes are costly and time-consuming.

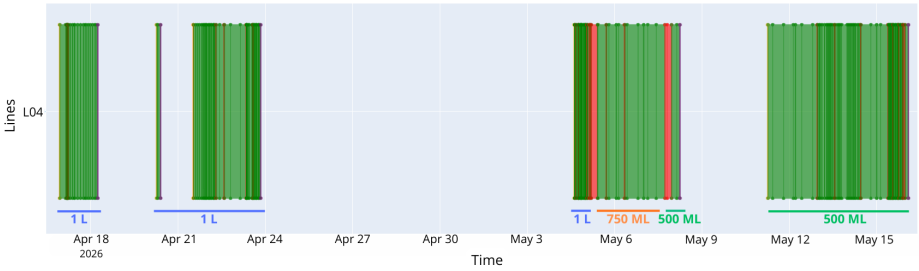


Fig. 5. Sequencing of line L04 in the baseline scenario, where all demands are assigned priority 10. The schedule reduces format changes.

To assess the impact of demand priorities, we consider a second scenario where we modify the priority of some delayed demands listed in Table 1 from $\pi_d = 10$ to $\pi_d = 7$. The resulting schedule for line L04 is shown in Figure 6.

Table 1. Demands assigned to line L04 whose priority is changed from $\pi_d = 10$ to 7.

Demand	**Format**	**Delivery date,** u_d	**ompletition date,** $\pi_d = 10$	**Completition date,** $\pi_d = 7$
d_1	1L	2026-04-04	2026-04-21	2026-04-17
d_2	1L	2026-03-30	2026-04-22	2026-04-17
d_3	1L	2026-04-04	2026-04-22	2026-04-17
d_4	500 ml	2026-03-30	2026-05-15	2026-04-22
d_5	500 ml	2026-04-28	2026-05-15	2026-04-22

Here, the algorithm modifies the production order to favor urgent demands. As a result, format 1L is produced in two separate runs: the first one includes priority 7 demands, but production is not extended further to avoid delaying the format 500 ml priority 7 demands. The latter is then produced in a more continuous run, while format 750 ml, which contains no priority 7 demands, is postponed to the end of the sequence. This is also reflected in Table 1, where demand completition dates are brought forward after increasing its priority.

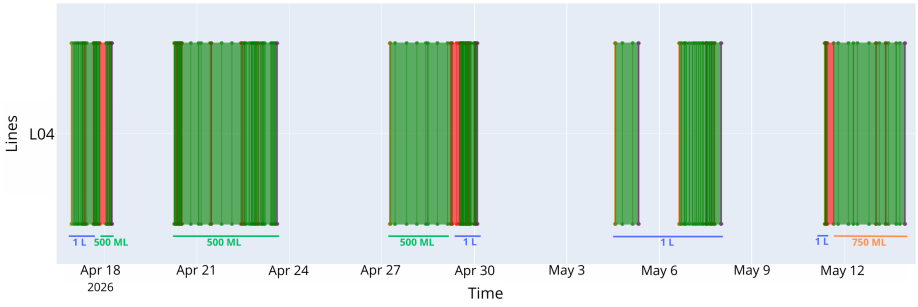


Fig. 6. Sequencing of line L04 after assigning priority 7 to some delayed demands of formats 1L and 500 ml. The sequence changes to protect urgent demands, even at the expense of splitting production runs.

Due to the industrial nature of the case study, a direct comparison with state-of-the-art methods is not possible, since available UPMSP benchmarks do not capture the full combination of proprietary constraints, lexicographic objectives and operational data considered. However, an indirect comparison with related metaheuristic approaches supports the proposed design: Ezugwu et al. report an average deviation of 3.57% for SOS-LPT, compared with 10.89% and 9.26% for

ACO-based baselines [3], while Yepes-Borrero et al. report RPD improvements above 70% and deviations below 6% on small UPMSR-S instances [8].

5 Conclusions

This paper presents some results of the Prodplanning project. The proposed heuristic framework preserves feasibility for an industrial UPMSP with tightly interdependent planning constraints. Its main value lies in maintaining feasibility throughout the construction and improvement phases, allowing the algorithm to provide realistic schedules for large instances and limited computational times.

The proposed method generates operationally meaningful plans and captures the expected trade-off between service level and production efficiency. In particular, the priority mechanism gives the planner control over which delayed demands should be protected, while the improvement phase can still exploit sequencing opportunities to reduce cost. The framework supports plant-dependent rules, making it suitable for heterogeneous industrial settings.

Future work will focus on providing a more detailed analysis of the improvement phase and extending the approach to dynamic re-planning.

6 Acknowledgements

This work was partially supported by 8IIDE2400018, within the framework of OGA research and development activities in artificial intelligence and operational research.

References

1. Duran, E., Ozturk, C., O'Sullivan, B.: Exact and Heuristic Methods for Planning and Scheduling Collaborative Manufacturing Systems, pp. 53–68 (2024)
2. Durasević, M., Jakobovic, D.: Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey. Artificial Intelligence Review **56** (2022)
3. Ezugwu, A.E., Adeleke, O.J., Viriri, S.: Symbiotic organisms search algorithm for the unrelated parallel machines scheduling with sequence-dependent setup times. PLoS ONE **13**, e0200030 (2018)
4. Fuchigami, H.Y., Rangel, S.: A survey of case studies in production scheduling: Analysis and perspectives. Journal of Computational Science **25**, 425–436 (2018)
5. García, J.M., Smith, K., Lozano, S., Guerrero, F.: A comparison of GRASP and an exact method for solving a production and delivery scheduling problem. In: Hybrid Information Systems. pp. 431–447 (2002)
6. Parunak, H.V.D.: Characterizing the manufacturing scheduling problem. Journal of Manufacturing Systems **10**, 241–259 (1991)
7. Pascal, C., Panescu, D.: On rescheduling in holonic manufacturing systems. Computers in Industry **104**, 34–46 (2019)
8. Yepes-Borrero, J.C., Alcaraz, J., López-García, M., Villaizán-Vallelado, M.: Unrelated parallel machine scheduling problem with setup times and additional resources: an enhanced metaheuristic to address resource-related infeasibilities. Annals of Operations Research **360**, 1149–1176 (2026)